IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR U.S. LETTERS PATENT

Title:

CACHING IN A VIRTUALIZATION SYSTEM

Inventors:

Robert L. Horn and Virgil V. Wilkins

Stephen A. Soffen
DICKSTEIN SHAPIRO MORIN &
OSHINSKY LLP
2101 L Street NW
Washington, DC 20037-1526
(202) 828-4879

CACHING IN A VIRTUALIZATION SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/426,822, filed November 18, 2002, and U.S. Provisional Application No. 60/505,023, filed September 24, 2003, the entire contents of which are incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates to a networked storage system with virtualization elements that contain cache.

BACKGROUND OF THE INVENTION

[0003] With the accelerating growth of Internet and intranet communication, high-bandwidth applications (such as streaming video), and large information databases, the need for networked storage systems has increased dramatically. One networked storage system architecture, the storage area network (SAN), provides a highly scalable, flexible topology that many experts are calling the future of enterprise storage.

[0004] In a SAN, users access the data on the storage elements through host ports. The host ports may be located in close proximity to the storage elements or they may be several miles away. In either case, the connection between the storage element controllers and the host ports is known as the SAN fabric. This fabric is often composed of a fiber channel interconnect, although, it may be any type of serial interconnect.

[0005] The storage elements used in SANs are often hard disk drives. Unfortunately, when a drive fails, the data stored on the drive is inaccessible. In a system where access to data is imperative, there must be a backup system. Most backup systems today involve storing the data on multiple disk drives so that, if one drive fails, another drive that contains a copy of the data is available. These multiple disk drives are known as

redundant arrays of independent disks (RAIDs). The addition of RAIDs and their associated RAID controllers make a SAN more reliable and fault tolerant. Because of its inherent advantages, RAID has quickly become an industry standard. However, there are still large groups of disk drives available for networked storage without the RAID features. These groups of disk drives are now referred to as "just a bunch of disks" (JBOD) to distinguish them from their RAID counterparts.

[0006] Storage systems often employ the use of several storage devices to redundantly store data (e.g., mirroring) in case one or more storage devices fail. Mirroring is a form of RAID known as RAID 1. Mirroring is the process by which data stored on one drive is copied or mirrored to another drive; therefore, the two drives are exact copies or mirrors of each other. In a like manner, several storage devices may be used in parallel to increase performance (striping). Striping is another aspect of RAID and is the process of breaking up consecutive lines of data and writing them on more than one drive. When the data needs to be accessed, all of the drives that contain a piece of the data may simultaneously send their portion to the requesting controller. The controller then arranges the data from each of the drives in order and sends it to the requesting host. However, it is inefficient for hosts to be required to keep track of the various logical and physical combinations, so a layer of abstraction is needed. This layer of abstraction is the concept of storage virtualization. Storage virtualization hides the internal functions of a storage subsystem or service from applications, computer servers, or general network resources for the purpose of enabling application and network independence from the management of storage or data. In a virtualized SAN architecture, hosts request access to virtual volumes, which may consist of any number of storage elements controlled by any number of RAID controllers. This allows for much greater flexibility in storage resource management, and allows volume size, performance, and reliability to change as users' needs change.

[0007] The virtualization layer is usually formed of virtualizer elements whose function is to translate virtual volume requests into logical volume requests and send those requests to the corresponding storage controllers. This process, of course, takes

some amount of overhead in the form of processing time. Processing cycles are required to translate the virtual addresses to their logical forms. Virtualizers also account for increased system latency because they constitute another layer of additional processing.

[0008] Still other problems with today's virtualizers include excessive interconnect traffic. Interconnect traffic includes data flowing to and from the disks, controllers, and virtualizers. In some cases, excessive interconnect traffic may occur when redundant data is sent over the interconnect multiple times. For example, a storage controller may send data to a disk it controls and send the same data to another controller that, in turn, sends the data to a disk under its control. The same data has now traversed the interconnect twice. Excessive interconnect traffic may limit the interconnect bandwidth and cause system performance to decrease. Thus there is a need for improved virtualization implementation in a networked storage system that reduces command latencies.

[0009] An example of a method for improving command latencies is described in U.S. Application Publication No. 2003/0084252, entitled, "Disk Drive Employing Adaptive Flushing of a Write Cache." The '252 application describes a method embodied as software or firmware code that permits the adaptation of disk drives employing write-back caching to reduce the possibility of lost data from the write cache. In one embodiment, the method is integrated with the host operating system software employed by a host computer coupled to the disk drive. The method issues write requests to the disk drive as it receives them from the applications running on the host computer. The disk drive processes the issued requests as it is designed to, using write-back caching techniques. After each request is cached, the disk drive controller acknowledges the write request back to the host. The host delays communicating the acknowledgements back to their originating applications until the data has been actually written to the disk media. Because write-back caching does not commit cached requests to disk on a regular basis, the host software simply forces the disk drive to execute cached write requests on a regular basis using a CACHE_FLUSH command. The disk drive employs standard throughput optimization techniques to reduce the overall latency of the disk accesses. When the rate of the request stream is low, the host simply issues a flush command after issuing each write

request to the drive. As the rate of the request stream increases, the host lets the requests pool in the cache rather than at the host. It then issues a flush command when the pool size reaches a number where the incremental reduction in throughput to the disk media during the flush no longer offsets the incremental increase in request latency due to the pooling time. When the flush is complete, the disk drive notifies the host, and the host releases the acknowledgements of all of the pooled requests to their originating applications.

[0010] The system described in the '252 application focuses on reducing latency and maintaining data integrity in a networked storage system, such as a SAN for write commands and write data. Although the '252 application describes a method of using write caching and acknowledging back to the host for increased system performance, it does not describe how to increase SAN performance for read commands. Furthermore, it does not describe a method for using cache in a networked storage virtualization layer. The system described in the '252 application also fails to provide a description of the virtualization process and how it may be produced or created.

SUMMARY OF THE INVENTION

[0011] The present invention is a scalable networked storage controller architecture that provides virtualization with cache for performing predictive reads and coalesced writes. The invention also provides an architecture that promotes reduction in latency and increased read-ahead efficiency in a storage area networks (SAN).

[0012] The present invention is a virtualizer and a method for operating the virtualizer. The virtualizer includes a target port for receiving primary data commands from a host system, a task manager for accepting primary data commands from the target port and coordinating execution of the primary data commands, a cache subsystem for receiving data requests corresponding to the primary data commands and reconciling the data requests, a command mapper for parsing the data requests into at least one secondary data command, and an initiator port for accepting the at least one secondary data

command and forwarding the at least one secondary data command to a downstream data storage element.

[0013] The method of operating the virtualizer includes the steps of receiving, via a target port, a primary data command from an external host system; forwarding the primary data command to a task manager; coordinating, in the task manager, execution of the primary data command at one of a host level and a volume-task set level; forwarding a data request corresponding to the primary data command to a cache subsystem, the cache subsystem reconciling the data request with a current state of the cache subsystem; retrieving data from the cache subsystem and forwarding the retrieved data to the target port, if the cache subsystem has the requested data; forwarding the data request to a command mapper, if the cache subsystem does not have the requested data; parsing of the data request into at least one secondary data command; forwarding the secondary data command to an initiator port; and forwarding the secondary data command to a downstream data storage element.

[0014] Therefore, it is an object of the present invention to reduce command latency and increase command throughput in a virtualization network through the incorporation of cache in a virtualizer.

[0015] It is another object of the invention to provide a virtualizer with the ability to perform predictive reads and coalesced writes through the incorporation of cache.

[0016] It is yet another object of this invention to enable RAID and/or JBOD controller functionality through the incorporation of cache in a virtualizer.

[0017] It is yet another object of this invention to enable RAID and/or JBOD controller functionality through the incorporation of cache in a virtualizer with reduction in latency.

[0018] It is yet another object of this invention to enable RAID and/or JBOD controller functionality through the incorporation of cache in a virtualizer with the ability to dictate predictive reads to the disk drives.

[0019] It is yet another object of this invention to enable RAID and/or JBOD controller functionality through the incorporation of cache in a virtualizer with the ability to perform coalesced writes.

[0020] It is yet another object of this invention to enable RAID and/or JBOD controller functionality through the incorporation of cache in a virtualizer with greater read-ahead efficiency.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] Figure 1 shows a block diagram of a virtualizer module that contains cache.

[0022] Figure 2 shows a block diagram of a virtualizer architecture with RAID controllers.

[0023] Figure 3 illustrates a block diagram of a virtualizer architecture with a JBOD and a RAID.

[0024] Figure 4 is a block diagram of the virtualizer architecture with an interconnect fabric.

DETAILED DESCRIPTION OF THE INVENTION

[0025] Figure 1 depicts a virtualizer module 100 in accordance with the invention. Virtualizer module 100 includes a target port 110, a command mapper 120, a task manager 130, a cache subsystem 140, and an initiator port 150.

[0026] The virtualizer module of the present invention may be implemented in hardware, software, firmware, Application Specification Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), Reduced Instruction Set Computers (RISCs) or any equivalent or combination thereof.

[0027] The elements of virtualizer module 100 are functionally connected as follows: Target port 110 is the data and control interface to an external host system (not

shown). Within virtualizer module 100, target port 110 is connected to task manager 130 and cache subsystem 140 through bi-directional busses. Cache subsystem 140 is a standard computer memory device that contains sub-components such as a prediction unit (not shown), a prefetch unit (not shown), a cache controller (not shown), and cache memory (not shown), as is well known to those skilled in the art. Task manager 130 is a digital control function that processes primary data commands received from an external host system and communicates with cache subsystem 140. Cache subsystem 140 is further connected within virtualizer module 100 to command mapper 120, which forwards additional/secondary data commands to initiator port 150. Initiator port 150 is connected within virtualizer module 100 to command mapper 120 and cache subsystem 140. Initiator port 150 is a physical port that externally connects to data storage elements (not shown) or to data storage element controllers (not shown) for the purpose of information storage and retrieval. Target port 110 and initiator port 150 are shown in Figure 1 as two physically distinct ports; however, they can, in fact, be the same physical port. It should be noted that the virtualizer module 100 is intended to illustrate a single, simple implementation of the invention. Those skilled in the art will recognize that a broad variety of other implementations, in accordance with the present invention, are also possible. For example, a virtualizer module may be equipped with multiple initiator ports (not shown) for the purpose of interfacing to multiple downstream data storage elements.

[0028] With reference to Figure 1, the essential function of virtualizer module 100 is generally described as follows: Target port 110 receives primary data command from an external host system (not shown) and forwards it to task manager 130. Task manager 130 coordinates the primary command execution at the host or volume-task set level. Task manager 130 then forwards user data requests directly to cache subsystem 140, which reconciles the user data request with the current cache state. If cache subsystem 140 can service the data request, cache data passes directly from cache subsystem 140 to target port 110. User data requests that cannot be serviced by the current cache state are forwarded by task manager 130 directly to command mapper 120. Command mapper 120 parses the user data request into one or more secondary data commands and forwards the secondary data commands directly to initiator port 150. Initiator port 150

then forwards these secondary data commands to downstream storage element/sub-systems.

[0029] Figure 2 is a virtualizer architecture with RAID 200 that includes a host 1 210 connected to a virtualizer module 1 220 and a virtualizer module 2 230 through a host interconnect 240. Virtualizer module 1 220 is also coupled to virtualizer module 2 230 via virtualizer interconnect 250. Virtualizer module 2 230 is further coupled to a host 1 210 via host interconnect 240 and a RAID controller 1 260 via virtualizer interconnect 250. RAID controller 1 260 is further coupled to a storage element 295 and a RAID controller 2 270. RAID controller 1 260 and storage element 295 communicate via a storage element interconnect 290. RAID controller 2 270 is similarly connected to storage element 295 via another storage element interconnect 291. RAID controller 1 260 and RAID controller 2 270 communicate via an interconnect 280. The use of multiple RAID controllers 260, 270 enhances system reliability. For example, host 1 210 issues a write command for a volume controlled by RAID controller 1 260 that resides on storage element 295. Since RAID controller 1 260 is redundantly paired with RAID controller 2 270, if RAID controller 1 260 fails, RAID controller 2 270 may take over control of storage element 295 because both RAID controllers 260, 270 are coupled to the storage element 295. Virtualizer architecture with RAID 200 reduces latency in the system because it reduces the number of steps required to give command completion status to the host. Virtualizer module 2 230 receives the write command from host 1 210. Virtualizer module 2 230 accepts the write data, stores it into its cache, and copies the data into the cache of virtualizer module 1 220 via virtualizer interconnect 250. Virtualizer module 1 220 acknowledges to virtualizer module 2 230 that the write data has been stored in cache. Virtualizer module 2 230 then acknowledges the write to host 1 210. At a later time, virtualizer module 2 230 forwards the write data with a write command to RAID controller 1 260. When the data has been written, RAID controller 1 260 sends an acknowledgement back to virtualizer module 2 230.

[0030] In contrast, a traditional system that contains no cache in the virtualizer modules must accept the write command and data from host 1 210 and forward

the command and write data to RAID controller 1 260. RAID controller 1 260 then copies the data to RAID controller 2 270. RAID controller 2 270 acknowledges to RAID controller 1 260 that the data is copied. RAID controller 1 260 further acknowledges the command to virtualizer module 2 230, which, in turn, acknowledges the write completion to host 1 210. In this case, the data is transferred from virtualizer module 2 230 to RAID controller 1 260 to RAID controller 2 270. In the present invention, the data is transferred using virtualizer architecture with RAID 200. Virtualizer architecture with RAID 200 thus provides less latency than conventional architectures because conventional systems require the RAID controller to decode the command, accept the command, mirror the command, and then acknowledge that it has received and mirrored the command back to the virtualizer. In turn, the virtualizer then acknowledges to the host that the command is complete. In contrast, virtualizer architecture with RAID 200 stores the command in its cache, mirrors the cache and acknowledges to the host that the command is complete without introducing latency from the RAID controller.

[0031] Figure 3 is a virtualizer architecture with JBOD and RAID 300 that includes the elements of virtualizer architecture with RAID 200 as well as a JBOD 310 coupled to virtualizer module 1 220. Virtualizer architecture with JBOD and RAID 300 allows for coalesced writes to JBOD 310. A coalesced write is simply the process of collecting multiple write requests to a group of sequential or nearly sequential logical block addresses (LBAs) so that the data may be written with a single write command to sequential LBAs. This process minimizes tracking and seeking motions performed by the head which, in turn, minimizes the time required to perform the writes as well as minimizing the physical head motion. Minimizing head motion increases the longevity of JBOD 310 and thus increases the mean time between failures (MTBF). The following is an example of a coalesced write. The example is used for illustrative purposes only and in no way limits the actual implementation of virtualizer architecture with JBOD and RAID 300. In this example, host 1 210 issues a write command to an LBA residing on JBOD 310. Virtualizer module 1 220 receives the command and data, stores the write data in the cache of virtualizer module 2 230. Virtualizer module 220 then sends a write acknowledge back to host 1 210. Host 1 210 issues a read command from an address on storage

element 295. Next, host 1 210 issues another write command to the next sequential LBA residing on JBOD 310. Virtualizer module 1 220 also stores this data in the cache of virtualizer module 2 230 and sends an acknowledge back to host 1 210. Host 1 210 then performs a write to storage element 295. Finally, host 1 210 sends a third write command to JBOD 310 via virtualizer module 1 220. This command and data are also stored in cache, and virtualizer module 220 acknowledges the command to host 1 210. The cache of virtualizer module 1 220 now holds the data for three write commands that are to be written to three consecutive LBAs on JBOD 310. Virtualizer module 1 220 creates a single write command from the three original write commands and sends the command and data to JBOD 310. JBOD 310 performs the three writes as a single write command and sends the complete acknowledgement to virtualizer module 1 220. The result is not only less wear and tear on the head of JBOD 310 but also in a reduction in latency. Using virtualizer architecture with JBOD and RAID 300, JBOD 310 finds the beginning LBA using a seek operation and performs the write for all three write requests. In a traditional system, JBOD 310 would need to locate three different LBAs, and then write three separate sets of data using multiple disk accesses at separate times.

[0032] Figure 4 is a virtualizer architecture with interconnect fabric 400. In this architecture, JBOD 310 is replaced with a RAID controller 3 430 mirrored with a RAID controller 4 440. RAID controller 3 430 controls a storage element 460 and is coupled to RAID controller 4 440 via an interconnect 450. The RAID controllers are coupled to the virtualizer modules via an interconnect fabric 410 and a virtualization layer interconnect 420. The RAID controllers 260, 270, 430, 440 are respectively coupled to storage elements 295, 295, 460, 460 respectively via storage element interconnects 290, 291, 292, 293. The following example illustrates the advantages of cache in virtualizers for predictive read performance improvements. In this example, virtual volume 1 stripes a logical volume on storage element 460 controlled by RAID controller 3 430 and a logical volume on storage element 295 controlled by RAID controller 1 260. Virtualizer module 1 220 and virtualizer module 2 230 have read-caching and read-ahead functionality. Because of this added functionality in the virtualizers, the read-ahead function may be disabled in the RAID controllers. Virtualizer 1 220 may now perform read functions more

efficiently because it, rather than the individual RAID controllers, has control over the read-ahead and read-caching operations. For example, host 1 210 issues a read command to virtualizer module 1 220 for virtual volume 1. Virtualization module 1 220 recognizes that virtual volume 1 includes a stripe across a logical volume residing on storage element 460 and a logical volume residing on storage element 295. For this example, a small portion of the data requested by host 1 210 resides at the end of the stripe on storage element 460 and the majority of the data resides in the stripe on storage element 295. Therefore, virtualizer module 1 220 issues a read command to RAID controller 3 430 for the data on storage element 460 and then issues either a larger command or a second and third command to RAID controller 1 260 for the rest of the data residing on storage element 295. In this manner, virtualizer module 1 220 has eliminated unnecessary read-ahead and read-caching that may have otherwise been performed by RAID controller 3 430. RAID controller 3 430 may have read-ahead data outside the stripe boundary, which would have been unnecessary and possibly detrimental to the life of storage element 460. Therefore, adding read-caching and read-ahead capability to the virtualizer modules improves the efficiency and the robustness of the overall system.

[0033] While the invention has been described and illustrated with reference to specific exemplary embodiments, it should be understood that many modifications and substitutions can be made without departing from the spirit and scope of the invention. Accordingly, the invention is not to be considered as limited by the foregoing description but is only limited by the scope of the appended claims.